6. Simple HTTP Request

Create a Ruby function `get_request(url)` that performs an HTTP GET request to the specified url and prints the response status code and body the output should be as shown bellow in json Format

Method : `get_request(url)`

```
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ cat 6-main.rb
require_relative '6-get'

get_request('http://jsonplaceholder.typicode.com/posts/1')
```

```
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 6-main.rb
Response status: 200 OK
Response body:
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
}
```

Implementation

```
require 'net/http'
require 'json'

def get_request(url)
  # Parse the URL
  uri = URI(url)

  # Perform the HTTP GET request
  response = Net::HTTP.get_response(uri)

  # Extract the status code and status message
  status_code = response.code
```

```ruby
  status_message = response.message

  # Extract and parse the response body
  response_body = JSON.parse(response.body)

  # Print the output in the specified format
  puts "Response status: #{status_code} #{status_message}"
  puts "Response body:"
  puts JSON.pretty_generate(response_body)
end
```

Explanation

1. **Require Necessary Libraries**:
    - `net/http`: Handles the HTTP request.
    - `json`: Parses and formats JSON responses.
2. **Parse the URL**:

    ```ruby
    uri = URI(url)
    ```

    Converts the provided URL into a URI object for the HTTP request.
3. **Perform HTTP GET Request**:

    ```ruby
    response = Net::HTTP.get_response(uri)
    ```

    Sends an HTTP GET request to the server and captures the response.
4. **Extract Status Code and Message**:

    ```ruby
    status_code = response.code
    status_message = response.message
    ```

    Retrieves the status code (e.g., `200`) and message (e.g., `OK`) from the response.
5. **Parse the Response Body**:

    ```ruby
    response_body = JSON.parse(response.body)
    ```

    Parses the response body (JSON string) into a Ruby hash for formatting.
6. **Output the Response**:

    ```ruby
    puts "Response status: #{status_code} #{status_message}"
    puts "Response body:"
    puts JSON.pretty_generate(response_body)
    ```

    - Prints the status code and message.
    - Formats the JSON body in a human-readable format using `JSON.pretty_generate`.

Test Case

Input:

```
get_request('http://jsonplaceholder.typicode.com/posts/1')
```

Output:

```
Response status: 200 OK
Response body:
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
}
```

Notes

- This implementation assumes the response body is always valid JSON. If the server returns non-JSON data, it may raise a `JSON::ParserError`. To handle this, you can add error handling around `JSON.parse`.
- The `Net::HTTP` library is included in the Ruby standard library, so you don’t need to install additional gems.